1

SQL Server 2008 Architecture

The days of SQL Server being merely a departmental database are long gone. SQL Server can now easily scale to databases dozens of terabytes in size. (For details see the results of the Winter survey at `www.microsoft.com/sql/prodinfo/compare/wintercorp-survey.mspx`.) In this chapter, we lay some of the groundwork that will be used throughout the book. We first discuss how the role of the database administrator (DBA) has changed since some of the earlier releases of SQL Server, and then quickly jump into architecture and tools available to you as an administrator. This chapter is not a deep dive into the architecture but it provides enough information to give you an understanding of how SQL Server operates.

The Expanding Role of a DBA

The role of the database administrator has been changing slowly over the past few versions of the SQL Server product. Beginning with SQL Server 2005, this slow transition of the DBA role has been accelerated immensely. Traditionally, a DBA would fit into one of two roles: development or administration. It's much tougher to draw a line now between DBA roles in SQL Server 2008. In addition, the new role of Business Intelligence DBA is on the rise. As lines blur and morph, DBAs have to quickly prepare themselves to take on different roles. If you don't position yourself to be more versatile, you may be destined for a career of watching SQL Server alerts and backups.

Production DBA

Production DBAs fall into the traditional role of a DBA. They are a company's insurance policy that the production database won't go down. If the database does go down, the company cashes

in its insurance policy in exchange for a recovered database. The Production DBA also ensures that the server is performing optimally, and he or she promotes database changes from development to quality assurance (QA) to production. Other tasks performed by a Production DBA include the following:

- Install SQL Server instances and service packs.
- Monitor performance problems.
- Install scripts from development.
- Create baselines of performance metrics.
- Configure the SQL Server optimally.
- Configure/implement high availability plans.
- Create\implement disaster recovery and scalability plans.
- Ensure that backups have been run.

Since the release of SQL Server 2000, there has been a trend away from full-time Production DBAs, and the role has merged with that of the Development DBA. The trend may have slowed, though, with laws such as Sarbanes-Oxley, which require a separation of power between the person developing the change and the person implementing the change. In a large organization, a Production DBA may fall into the operations department, which would consist of the network administrators and Windows-support administrators. Placing a Production DBA in a development group removes the separation of power that may be needed for some regulatory reasons. It may create an environment where ''rush'' changes are immediately put into production, without proper inspection and auditing.

Development DBA

Development DBAs also play a very traditional role in an organization. They wear more of a developer's hat and are the development staff's database experts and representatives. This administrator ensures that all stored procedures are optimally written and that the database is modeled correctly, both physically and logically. He or she also may be the person who writes the migration processes to upgrade the database from one release to the next. The Development DBA typically does not receive calls at 2:00 A.M. Other Development DBA tasks may be as follows:

- Model an application database.
- Create stored procedures.
- Develop the change scripts that go to the Production DBA.
- Performance-tune queries and stored procedures.
- Create data migration plans and scripts.
- Serve as an escalation point for the Production DBA.

The Development DBA typically would report to the development group. He or she would receive requests from a business analyst or another developer. In a traditional sense, Development DBAs should

never have modification access to a production database. They should, however, have read-only access to the production database to debug in a time of escalation.

Business Intelligence DBA

The Business Intelligence (BI) DBA is a new role that has evolved due to the increased capabilities of SQL Server. In SQL Server 2005, BI grew to be an incredibly important feature set that many businesses could not live without. The BI DBA is an expert at these features.

BI DBAs may have specializations, just like normal SQL DBAs. A Production BI DBA will perform the same functions as the Production DBA: installs, service packs, deployments, high availability, performance tuning, and backups. The only difference is that the Production BI DBA will be paying closer attention to SQL Server Analysis Services (SSAS), SQL Server Integration Services (SSIS), SQL Server Reporting Services (SSRS), and perhaps Proclarity, Business Scorecard Manager, and Performance Point Servers.

Development BI DBAs specialize in the best practices, optimization, and use of the BI toolset. In a small organization, he or she may create your SSIS packages to perform Extract Transform and Load (ETL) processes or reports for users. In a large organization, developers create the SSIS packages and SSRS reports. The Development BI DBA is consulted regarding the physical implementation of the SSIS packages, and Analysis Services (SSAS) cubes. Development BI DBAs may be responsible for the following types of functions:

- Model\consult regardingAnalysis Services cubes and solutions.
- Create reports using Reporting Services.
- Create\consult around ETL using Integration Services.
- Develop deployment packages that will be sent to the Production DBA.

Organizationally, the BI DBA most often reports to the development group. In some cases, Analysis Services experts may report to the analyst group or the project management office. In some small organizations, the BI DBA may report directly to an executive such as a CFO.

Hybrid DBA

The most exciting role for a DBA is a hybrid of all the roles just mentioned. This Hybrid DBA is very typical with smaller organizations but is becoming popular with larger organizations as well. An organization with high turnover may want to spread its investment over many Hybrid DBAs instead of relying on specialized roles.

Organizationally, you may see Hybrid DBAs reporting directly to the product organization or to a specialized DBA group. No matter where these DBAs report, each typically has a slate of products that he or she supports, performing every DBA function for that product. Organizations that rely on Hybrid DBAs should have adequate backup personnel to reduce the organization's risk if a Hybrid DBA leaves the company. Also, this DBA should never install his or her own changes into production.

Ideally, for regulatory reasons and for stability, the DBA's backup DBA should install the change into production. That way, you can ensure that the DBA who installed the script didn't make ad hoc changes in order to make the change work. We cover much more about this change-management process in Chapter 10.

The only role of a Hybrid DBA that's questionable is development of stored procedures. In most organizations where we see this role, the Hybrid DBA does not develop stored procedures. Instead, he or she creates difficult stored procedures or tunes the ones causing issues. The developer working on the application develops his or her own stored procedures and then provides them to the Hybrid DBA to package and proof. The main reason for this is that the DBA is too taxed for time, working on other functions of the database.

New Things You Need to Learn

The best of us continue to learn new skills and keep up with the changing face of software. It is the business we are in. We must continue to grow and learn or risk becoming obsolete. Each new release of SQL Server since 7.0 has required DBAs to know more things that were traditional concerns of developers. As Microsoft puts more and more on the SQL Server CD, and integrates SQL Server with other development environments, programs, and tools, the breadth of our skills must also grow. Here are some reminders of items that warrant your attention:

- Resource Governor allows you to manage workload by setting resource limits. New in SQL Server 2008, knowledge of this feature is a must for DBAs.
- Certificates and Kerberos have been used in SQL Server since SQL 2005. While you do not need to be an expert, you must spend some time getting acquainted with how these things work. Kerberos will become especially important if your site uses Reporting Services (SSRS) and your Reporting Services database is on a separate server than Reporting Services Web Service.
- CLR Integration enables you to use .NET programming in your stored procedures, triggers, and functions. It also means you need to learn a .NET programming language, or at least the basics of one. You should become acclimated to a .NET programming language such as C# or VB.NET to remain effective. For example, if you are a DBA trying to debug a performance problem with a CLR stored procedure, then you need to know the language the stored procedure is written in to understand the performance problem. Features such as Integration Services and Reporting Services are very much tied to expressions, which are variants of VB.NET.
- You need to learn something about XML, including some XPath, and XQuery. These features, introduced in SQL Server 2005, are now getting use in some implementations.
- Get some practical experience on database mirroring.
- Of course, you should learn about SSRS, SSIS, and SSAS, even if your shop does not currently use those features.

Beginning with SQL Server 2005 and continuing for SQL Server 2008, these products require a leap forward in the knowledge a DBA must have to be effective. If you want to be a leader, then you must stay ahead of the game. We'll help you get it done.

SQL Server Architecture

In older editions of SQL Server, you had to use many different tools depending on the function you were trying to perform. In SQL Server 2008, the challenge for Microsoft was to avoid increasing the number of management tools while increasing the features and products that ship with SQL Server. They accomplished this by creating one tool for business-intelligence development (Business Intelligence Development Studio — BIDS) and another for management of the entire platform, including business intelligence and the database engine (SQL Server Management Studio). BIDS is based on a lightweight version of Visual Studio 2008. A new end-user report development tool is also added — Report Designer.

SQL Server envelops a large surface now. It can act as a reporting tool and store your OLAP cubes. It can also perform your ETL services through SQL Server Integration Services. Many people just use SQL Server for its classic use: to store data. SQL Server 2008 can run on Windows XP, 2000, Vista, and Windows Server 2003 and 2008. Tools such as SharePoint and Office quickly integrate on top of SQL Server and can provide an easy user interface (UI) for SQL Server data. This book covers administration on each of these tiers.

Transaction Log and Database Files

The architecture of database and transaction log files remains unchanged from prior releases. The purpose of the transaction log is to ensure that all committed transactions will be persisted in the database and can be recovered.

The transaction log is a *write-ahead* log. As you make changes to a database in SQL Server, the record is first written to the transaction log. Then, during a checkpoint and at other times, the log data is quickly transferred to the data file. This is why you may see your transaction log grow significantly in the middle of a long-running transaction even if your recovery model is set to simple. (We cover this in much more detail in Chapter 18.)

Every time SQL Server starts, it performs a recovery process on each database. The recovery process ensures that the data in the database is in a consistent state. This means that all committed transactions are recorded in the data files, and that no uncommitted data is in the data files. The recovery process reads the transaction log, looking for any committed transactions that were never added to the data file. The recovery process adds this data to the data file. This is called *rolling a transaction forward*. Recovery also looks for any uncommitted changes that may have been pre-written to the data files. Because the transaction did not commit, recovery will remove these changes from the data files. This is called *rolling a transaction back*. In SQL Server 2008 Enterprise Edition, this process can be done in parallel across all the databases on your instance. Additionally, a fast recovery feature in Enterprise Edition makes databases available after the roll-forward process is complete.

> *The recovery process also runs at the end of a restore. Although there is some confusion and misuse of terms, even in Microsoft's Books Online,* `Restore` *replaces a database from backups. This only occurs when you use the* `Restore` *T-SQL command. The recovery process runs at the end of the restore and during startup, to ensure that the database is in a consistent state.*

A database may consist of multiple filegroups. Each filegroup may contain one or more physical data files. Filegroups are used to ease administrative tasks for a collection of files. Data files are divided into

8KB data pages. You can specify how full each data page should be with the fill factor option of the `create/alter index` T-SQL command. (We go much more into this in Chapter 14.) In SQL Server 2008, you have the capability to bring your database partially online if a single file is corrupt. In this instance, the DBA can bring the remaining files online for reading and writing, and the user receives an error if he or she tries to access the other parts of the database that are offline. (You'll learn much more about this in Chapter 18.)

Historically, the largest row you could write has been 8060 bytes. There are two exceptions to this limit: `text`, `ntext`, `image`, `varchar(max)`, `varbinary(max)`, and `nvarchar(max)` columns may each be up to 2 gigabytes large, and are managed separately. Beginning with SQL 2005, the 8KB limit applies only to those columns of fixed length. The sum of fixed-length columns, and pointers for other column types, must still be less than 8060 bytes per row. However, each variable-length column may be up to 8KB in size, so the row size can be larger than 8KB in total. If your actual row size exceeds 8060 bytes, you may experience some performance degradation, as the logical row must now be split across multiple physical 8060-byte rows.

SQL Native Client

The SQL Native Client is a data-access method that ships with SQL Server 2008 and is used by both OLE DB and ODBC for accessing SQL Server. The SQL Native Client simplifies access to SQL Server by combining the OLE DB and ODBC libraries into a single access method. The access type exposes some of the new features in SQL Server:

- Database mirroring
- Multiple Active Recordsets (MARS)
- Snapshot isolation
- Query notification
- XML data type support
- User-defined data types (UDTs)
- Encryption
- Performing asynchronous operations
- Using large value types
- Performing bulk copy operations
- Table-value parameters
- Large CLR user-defined types
- Password expiration

In some of these features, you can use the feature in other data layers such as Microsoft Data Access Components (MDAC), but it will take more work. MDAC still exists, and you can use it if you don't need some of the new functionality of SQL Server 2005\2008. If you are developing a COM-based application, you should use SQL Native Client; and if you are developing a managed code application like in C#, you should consider using the .NET Framework Data Provider for SQL Server, which is very robust and includes the SQL Server 2005\2008 features as well.

System Databases

The system databases in SQL Server are crucial, and you should leave them alone most of the time. The only exception to that rule is the `model` database, which allows you to deploy a change such as a stored procedure to any new database created.

> **If a system database is tampered with or corrupted, you run the risk that SQL Server will not start. It contains all the stored procedures and tables needed for SQL Server to remain online.**

The Resource Database

SQL Server 2005 added the `Resource` database. This database contains all the read-only critical system tables, metadata, and stored procedures that SQL Server needs to run. It does not contain any information about your instance or your databases, because it is only written to during an installation of a new service pack. The `Resource` database contains all the physical tables and stored procedures referenced logically by other databases. The database can be found by default in `C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Binn.mdf and .ldf`, and there is only one `Resource` database per instance.

> *The use of drive C: in the path assumes a standard setup. If your machine is set up differently, you may need to change the path to match your setup. Additionally, the .MSSQLSERVER is the instance name. If your instance name is different, use your instance name in the path.*

In SQL Server 2000, when you upgraded to a new service pack, you would need to run many long scripts to drop and recreate system objects. This process took a long time to run and created an environment that couldn't be rolled back to the previous release after the service pack. In SQL Server 2008, when you upgrade to a new service pack or quick fix, a copy of the `Resource` database overwrites the old database. This enables you to both quickly upgrade your SQL Server catalog and roll back a release.

The `Resource` database cannot be seen through Management Studio and should never be altered unless you're under instruction to do so by Microsoft Product Support Services (PSS). You can connect to the database under certain single-user mode conditions by typing the command `USE MSSQLSystemResource`. Typically, a DBA runs simple queries against it while connected to any database, instead of having to connect to the resource database directly. Microsoft provides some functions which allow this access. For example, if you were to run this query while connected to any database, it would return your `Resource` database's version and the last time it was upgraded:

```
SELECT serverproperty('resourceversion') ResourceDBVersion,
serverproperty('resourcelastupdatedatetime') LastUpdateDate
```

> *Do not place the* `Resource` *database on an encrypted or compressed drive. Doing this may cause upgrade or performance issues.*

The master Database

The `master` database contains the metadata about your databases (database configuration and file location), logins, and configuration information about the instance. You can see some of the metadata stored

in `master` by running the following query, which returns information about the databases that exist on the server:

```
SELECT * FROM sys.databases
```

The main difference between the `Resource` and `master` databases is that the `master` database holds data specific to your instance, whereas the `Resource` database just holds the schema and stored procedures needed to run your instance, but does not contain any data specific to your instance. You should always back up the `master` database after creating a new database, adding a login, or changing the configuration of the server.

> **You should never create objects in the `master` database. If you create objects here, you may need to make more frequent master db backups.**

tempdb Database

The `tempdb` database is similar to the operating system paging file. It's used to hold temporary objects created by users, temporary objects needed by the database engine, and row-version information. The `tempdb` database is created each time you restart SQL Server. The database will be recreated to be its original database size when the SQL Server is stopped. Because the database is recreated each time, there is no reason to back it up. Data changes made to objects in the `tempdb` database benefit from reduced logging. It is important to have enough space allocated to your `tempdb` database, because many operations that you will use in your database applications use the `tempdb`. Generally speaking, you should set `tempdb` to autogrow as it needs space. If there is not enough space, the user may receive one of the following errors:

- **1101** or **1105:** The session connecting to SQL Server must allocate space in `tempdb`.
- **3959:** The version store is full.
- **3967:** The version store must shrink because `tempdb` is full.

model Database

`model` is a system database that serves as a template when SQL Server creates a new database. As each database is created, SQL Server copies the `model` database as the new database. The only time this does not apply is when you restore or attach a database from a different server.

> *If a table, stored procedure, or database option should be included in each new database that you create on a server, you may simplify the process by creating the object in* `model`*. When the new database is created,* `model` *is copied as the new database, including the special objects or database settings you have added to the* `model` *database. If you add your own objects to* `model`*, model should be included in your backups, or you should maintain a script which includes the changes.*

msdb Database

`msdb` is a system database that contains information used by SQL Server agent, log shipping, SSIS, and the backup and restore system for the relational database engine. The database stores all the information about jobs, operators, alerts, and job history. Because it contains this important system-level data, you should back up this database regularly.

Schemas

Schemas enable you to group database objects together. You may wish to do this for ease of administration, as you can apply security to all objects within a schema. Another reason to use schemas is to organize objects so the consumers may find the objects they need easily. For example, you may create a schema called `HumanResource` and place all your employee tables and stored procedures into it. You could then apply security policies on the schema to allow appropriate access to the objects contained within it.

When you refer to an object you should always use the two-part name. The `dbo` schema is the default schema for a database. An `Employee` table in the `dbo` schema is referred to as `dbo.Employee`. Table names must be unique within a schema. You could create another table called `Employee` in the `HumanResources` schema. It would be referred to as `HumanResources.Employee`. This table actually exists in the `AdventureWorks2008` sample database for SQL Server 2008. (All SQL Server 2008 samples must be downloaded and installed separately.) A sample query using the two-part name follows:

```
SELECT BusinessEntityID, JobTitle
FROM HumanResources.Employee
```

Prior to SQL 2005, the first part of the two-part name was the user name of the object owner. The problem with that implementation was related to maintenance. If a user who owned objects was to leave the company, you could not remove that user login from SQL Server until you ensured that all the objects owned by the user were changed to a different owner. All of the code that referred to the objects had to be changed to refer to the new owner. By separating ownership from the schema name, SQL 2005 and 2008 remove this maintenance problem.

Synonyms

A *synonym* is an alias, or alternate name, for an object. This creates an abstraction layer between the database object and the consumer. This abstraction layer enables you to change some of the physical implementation, and isolate those changes from the consumer. An example is related to the use of linked servers. You may have tables on a different server which need to be joined to tables on a local server. You refer to objects on another server using the four-part name, as shown in the following code:

```
SELECT Column1, Column2
FROM LinkedServerName.DatabaseName.SchemaName.TableName
```

For example, you might create a synonym for `LinkedServerName.DatabaseName.SchemaName.Tablename` called `SchemaName.SynonymName`. Data consumers would refer to the object using the following query:

```
SELECT Column1, Column2
FROM SchemaName.SynonymName
```

This abstraction layer now enables you to change the location of the table to another server, using a different linked server name, or even to replicate the data to the local server for better performance without requiring any changes to the code which refers to the table.

A synonym cannot reference another synonym. The `object_id` *function returns the id of the synonym, not the id of the related base object. If you need column-level abstraction, use a view instead.*

Dynamic Management Views

Dynamic management views (DMVs) and functions return information about your SQL Server instance and the operating system. DMVs simplify access to data and expose new information that was not available in versions of SQL Server prior to 2005. DMVs can provide you with various types of information, from data about the I/O subsystem and RAM to information about Service Broker.

Whenever you start an instance, SQL Server begins saving server-state and diagnostic information into DMVs. When you stop and start the instance, the information is flushed from the views and fresh data begins to be loaded. You can query the views just like any other table in SQL Server with the two-part qualifier. For example, the following query uses the `sys.dm_exec_sessions` DMV to retrieve the number of sessions connected to the instance, grouped by login name:

```
SELECT login_name, COUNT(session_id) as NumberSessions
FROM sys.dm_exec_sessions GROUP BY login_name
```

Some DMVs are functions which accept parameters. For example, the following code uses the `sys.dm_io_virtual_file_stats` dynamic management function (we use the term DMV for simplicity throughout this book) to retrieve the I/O statistics for the AdventureWorks2008 data file:

```
SELECT * FROM
sys.dm_io_virtual_file_stats(DB_ID('AdventureWorks2008'),
FILE_ID('AdventureWorks2008_Data'))
```

You'll learn much more about DMVs throughout this book, starting in Chapter 4.

SQL Server 2008 Data Types

As you create a table, you must assign a data type for each column. In this section, we cover some of the more commonly used data types in SQL Server. Even if you create a custom data type, it must be based on a standard SQL Server data type. For example, you may create a custom data type (`Address`) by using the following syntax, but notice that it based on the SQL Server standard `varchar` data type:

```
CREATE TYPE Address
FROM varchar(35) NOT NULL
```

If you are changing the data type of a column in a very large table in SQL Server Management Studio's table designer interface, the operation may take a very long time. You can observe the reason for this by scripting the change from the Management Studio interface. Management Studio creates a secondary temporary table with a name like `tmpTableName` and then copies the data into the table. Finally, the interface deletes the old table and renames the new table with the new data type. There are other steps along the way, of course, to handle indexes and any relationships in the table.

If you have a very large table with millions of records, this process can take more than ten minutes, and in some cases more than an hour. To avoid this, you can use a simple one-line T-SQL statement in the query window to change the column's data type. For example, to change the data type of the Job `Title` column in the `Employees` table to a `varchar(70)`, you could use the following syntax:

```
ALTER TABLE HumanResources.Employee ALTER COLUMN JobTitle Varchar(70)
```

When you convert to a data type that may be incompatible with your data, you may lose important data. For example, if you convert from a numeric data type that has data such as 15.415 to an integer, the number 15.415 would be rounded to a whole number.

You may wish to write a report against your SQL Server tables which displays the data type of each column inside the table. There are dozens of ways to do this, but one method we often see is to join the `sys.objects` table with the `sys.columns` table. There are two functions that you may not be familiar with in the following code. The `TYPE_NAME()` function translates the data type id into its proper name. To go the opposite direction, you could use the `TYPE_ID()` function. The other function of note is `SCHEMA_ID()`, which is used to return the identity value for the schema. This is useful primarily when you wish to write reports against the SQL Server metadata.

```
SELECT o.name AS ObjectName,
       c.name AS ColumnName,
       TYPE_NAME(c.user_type_id) as DataType
FROM   sys.objects o JOIN sys.columns c
ON     o.object_id = c.object_id
WHERE  o.name ='Department'
and o.Schema_ID = SCHEMA_ID('HumanResources')
```

This code returns the following results (note that the `Name` data type is a user-defined type):

```
ObjectName           ColumnName     DataType
-----------------------------------------------------

Department           DepartmentID   smallint
Department           Name           Name
Department           GroupName      Name
Department           ModifiedDate   datetime
```

Character Data Types

Character data types include `varchar`, `char`, `nvarchar`, and `nchar`, `text`, and `ntext`. This set of data types stores character data. The primary difference between the `varchar` and `char` types is data padding. If you have a column called `FirstName` that is a `varchar(20)` data type and you store the value of ''Brian'' in the column, only five bytes will be physically stored. If you store the same value in a `char(20)` data type, all 20 bytes would be used. SQL will insert trailing spaces to fill the 20 characters.

If you're trying to conserve space, why would you ever use a `char` data type? There is a slight overhead to using a `varchar` data type. If you are going to store a two-letter state abbreviation, for example, you're better off using a `char(2)` column. Although some DBAs have opinions about this that border on religious conviction, generally speaking it's good to find a threshold in your organization and specify that anything below this size will become a `char` versus a `varchar`. Our guideline is that, in general, any column that is less than or equal to five bytes should be stored as a `char` data type instead of a `varchar` data type. Beyond that point, the benefit of using a `varchar` begins to outweigh the cost of the overhead.

The `nvarchar` and `nchar` data types operate the same way as their `varchar` and `char` counterparts, but these data types can handle international Unicode characters. This comes at a cost though. Data stored as

Unicode consumes 2 bytes per character. If you were to store the value of ''Brian'' in an `nvarchar` column, it would use 10 bytes, and storing it as an `nchar(20)` would use 40 bytes. Because of this overhead and added space, do not use Unicode columns unless you have a business or language need for them.

Next are `text` and `ntext`. The `text` data type stores very large character data on and off the data page. You should use these sparingly, as they may affect performance. They can store up to 2GB of data in a single row's column. Instead of using the `text` data type, the `varchar(max)` type is a much better alternative because the performance is better. Additionally, `text` and `ntext` data types will not be available in some future version of SQL Server, so begin using `varchar(max)` and `nvarchar(max)` instead of `text` and `ntext` now.

The following table shows the data types, with short descriptions and the amount of storage required.

Data Type	Description	Storage Space
`Char(n)`	N between 1 and 8,000 characters	n bytes
`Nchar(n)`	N between 1 and 4,000 Unicode characters	(2 x n bytes) + 2 bytes overhead
`Ntext`	Up to ((2 to the 30th power) - 1) (1,073,741,823) Unicode characters	2 bytes per character stored
`Nvarchar(max)`	Up to ((2 to the 30th power) - 1) (1,073,741,823) Unicode characters	2 x characters stored + 2 bytes overhead
`Text`	Up to ((2 to the 31st power) - 1) (2,147,483,647) characters	1 byte per character stored
`Varchar(n)`	N between 1 and 8,000 characters	1 byte per character stored + 2 bytes overhead
`Varchar(max)`	Up to ((2 to the 31st power) - 1) (2,147,483,647) characters	1 byte per character stored + 2 bytes overhead

Exact Numeric Data Types

Numeric data types consist of `bit`, `tinyint`, `smallint`, `int`, `bigint`, `numeric`, `decimal`, `money`, `float`, and `real`. Each of these data types stores different types of numeric values. The first data type, `bit`, stores only a 0 or a 1, which in most applications translates into true or false. Using the `bit` data type is perfect for on and off flags, and it occupies only a single byte of space. Other common numeric data types are shown in the following table.

Data Type	Description	Storage Space
`bit`	0, 1, or Null	1 byte for each 8 columns of this data type
`tinyint`	Whole numbers from 0 to 255	1 bytes
`smallint`	Whole numbers from –32,768 to 32,767	2 bytes
`int`	Whole numbers from –2,147,483,648 to 2,147,483,647	4 bytes

Data Type	Description	Storage Space
`bigint`	Whole numbers from –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 bytes
`numeric(p,s)` or `decimal(p,s)`	Numbers from –1,038 +1 through 1,038 –1	Up to 17 bytes
`money`	–922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
`smallmoney`	–214,748.3648 to 214,748.3647	4 bytes

Numeric data types, such as `decimal` and `numeric`, can store a variable number of digits to the right and left of the decimal place. *Scale* refers to the number of digits to the right of the decimal. *Precision* defines the total number of digits, including the digits to the right of the decimal place. For example, 14.88531 would be a `numeric(7,5)` or `decimal(7,5)`. If you were to insert 14.25 into a `numeric(5,1)` column, it would be rounded to 14.3.

Approximate Numeric Data Types

The data types `float` and `real` are included in this group. They should be used when floating-point data must be represented. However, because they are approximate, not all values can be represented exactly.

The `n` in the `float(n)` is the number of bits used to store the mantissa of the number. SQL Server uses only two values for this field. If you specify between 1 and 24, SQL uses 24. If you specify between 25 and 53, SQL uses 53. The default is 53 when you specify `float()`, with nothing in parenthesis.

The following table shows the approximate numeric data types, with a short description and the amount of storage required.

Data Type	Description	Storage Space
`float[(n)]`	−1.79E+308 to -2.23E-308,0, 2.23E-308 to 1.79E+308	N< = 24 - 4 bytes N> 24 - 8 bytes
`real()`	−3.40E+38 to -1.18E-38,0, 1.18E-38 to 3.40E+38	4 bytes

The synonym for `real` *is* `float(24)`.

Binary Data Types

Binary data types such as `varbinary`, `binary`, `varbinary(max)`, and `image` store binary data such as graphic files, Word documents, or MP3 files. The values are hexadecimal 0x0 to 0xf. The `image` data type stores up to 2GB outside the data page. The preferred alternative to an `image` data type is the

varbinary(max), which can hold more than 8KB of binary data and generally performs slightly better than an image data type. New in SQL Server 2008 is the capability to store varbinary(max) objects in operating system files via FileStream storage options. This option stores the data as files, and is not subject to the 2GB size limit of varbinary(max).

The following table shows the binary data types, with a short description and the amount of storage required.

Data Type	Description	Storage Space
Binary(n)	N between 1 and 8,000 hex digits	n bytes
Image	Up to 231-1(2,147,483,647) hex digits	1 byte per character
Varbinary(n)	N between 1 and 8,000 hex digits	1 byte per character stored + 2 bytes overhead
Varbinary(max)	Up to 231-1(2,147,483,647) characters	1 byte per character stored + 2 bytes overhead

Date and Time Data Types

The datetime and smalldatetime types both store date and time data. The smalldatetime is 4 bytes and stores from January 1, 1900 through June 6, 2079 and is accurate to the nearest minute. The datetime data type is 8 bytes and stores from January 1, 1753 through December 31, 9999 to the nearest 3.33 millisecond.

SQL Server 2008 has four new date-related data types: datetime2, dateoffset, date, and time. You can find examples using these data types in SQL Server Books Online.

The datetime2 data type is an extension of the datetime data type, with a wider range of dates. Time is always stored with hours, minutes, and seconds. You can define the datetime2 data type with a variable parameter at the end — for example, datetime2(3). The 3 in the preceding expression means to store fractions of seconds to three digits of precision, or .999. Valid values are between 0 and 9, with a default of 3.

The datetimeoffset data type is just like the datetime2 data type, with the addition of the time offset. The time offset is + or - up to 14 hours, and contains the UTC offset, so that you can rationalize times captured in different time zones.

The date data type stores the date only, a long-requested piece of functionality. Alternately, the time data type stores the time only. The time data type also supports the time(n) declaration so you can control granularity of the fractional seconds. As with datetime2 and datetimeoffset, n can be between 0 and 7.

The following table shows the date/time data types, with a short description and the amount of storage required.

Data Type	Description	Storage Space
`Date`	January 1, 1 to December 31, 9999	3 bytes
`Datetime`	January 1, 1753 to December 31, 9999, Accurate to nearest 3.33 millisecond	8 bytes
`Datetime2(n)`	January 1, 1 to December 31, 9999 N between 0 and 7 specifies fractional seconds	6 to 8 bytes
`Datetimeoffset(n)`	January 1, 1 to December 31, 9999 N between 0 and 7 specifies fractional seconds +- offset	8 to 10 bytes
`SmalldateTime`	January 1, 1900 to June 6, 2079, Accurate to 1 minute	4 bytes
`Time(n)`	Hours:minutes:seconds.9999999 N between 0 and 7 specifies fractional seconds	3 to 5 bytes

Other System Data Types

There are several other data types which we have not seen. They are shown in the following table for completeness.

Data Type	Description	Storage Space
`Cursor`	This contains a reference to a cursor and may be used only as a variable or stored procedure parameter.	Not applicable
`Hierarchyid`	Contains a reference to a location in a hierarchy	1 to 892 bytes + 2 bytes overhead
`SQL_Variant`	May contain the value of any system data type except `text`, `ntext`, `image`, `timestamp`, `xml`, `varchar(max)`, `nvarchar(max)`, `varbinary(max)`, `sql_variant`, and user-defined data types. The maximum size allowed is 8,000 bytes of data + 16 bytes or metadata.	8,016 bytes
`Table`	Used to store a data set for further processing. The definition is like a Create Table. Primarily used to return the result set of a table-valued function, they can also be used in stored procedures and batches.	Dependent on table definition and number of rows stored

Data Type	Description	Storage Space
`Timestamp` or `Rowversion`	Unique per table, automatically stored value. Generally used for version stamping, the value is automatically changed on insert and with each update.	`8 bytes`
`Uniqueidentifier`	Can contain Globally Unique Identifier (GUID). `guid` values may be obtained from the `Newid()` function. This function returns values which are unique across all computers. Although stored as a binary 16, it is displayed as a `char(36)`.	16 bytes
`XML`	Can be stored as Unicode or non-Unicode	Up to 2GB

A `cursor` *data type may not be used in a* `Create Table` *statement.*

The `hierarchyid` column is new to SQL Server 2008. You may wish to add a column of this data type to tables where the data in the rows can be represented in a hierarchy, as in an organizational hierarchy or manager/employee hierarchy. The value that you store in this column is the path of the row within the hierarchy. Levels in the hierarchy are shown as slashes. The value between the slashes is the numerical location of this member within the row. An example is `/1/3`. Special functions are available which can be used with this data type.

The `XML` data type stores an XML document or fragment. It is stored like a `text` or `ntext` in size depending on the use of UTF-16 or UTF-8 in the document. The `XML` data type allows the use of special constructs for searching and indexing. (This is covered in more detail in Chapter 15.)

CLR Integration

In SQL Server 2008, you can also create your own data types and stored procedures using the CLR (Common Language Runtime). This enables you to write more complex data types to meet your business needs in Visual Basic or C#, for example. These types are defined as a class structure in the base CLR language. (We cover the administrative aspect of these in much more detail in Chapter 8.)

Editions of SQL Server

SQL Server 2008 is available in numerous editions, and the features available to you in each edition vary widely. The editions you can install on your workstation or server also vary based on the operating system. The editions of SQL Server range from SQL Express on the lowest end to Enterprise Edition on the highest. The prices of these also vary widely, from free to more than $20,000 per processor.

Ted Kummert, Microsoft corporate vice president, announced at the Professional Association for SQL Server (PASS) conference in September, 2007, that prices for SQL Server 2008 would remain the same as they were for SQL 2005. No price increase — woohooo!

Compact (32-bit Only)

SQL Compact is a free edition which is intended to be an embedded database for mobile and other compact devices with occasionally connected users.

SQL Express (32-bit Only)

SQL Express is the free version of SQL Server meant for installation on laptops or desktops to support distributed applications such as a remote sales force application. You can use this edition to store sales or inventory data for your disconnected sales force and replicate updated data to them when they become connected again. SQL Express was called Microsoft Desktop Edition (MSDE) in SQL Server 2000. It is extremely lightweight and does not occupy much hard drive space. Vendors are free to distribute SQL Express, and it can be wrapped into your application's installation as just another component.

SQL Express is not meant to scale past a few users. Key features missing from SQL Express are SQL Agent and some of the robust management tools. It does ship with a very lightweight tool for managing the database, but scheduling of backups must be done in the Windows Task Scheduler, not SQL Server.

Workgroup Edition (32-bit and 64-bit)

The Workgroup Edition of SQL Server is the lowest-cost commercial edition of SQL Server. It scales minimally up to two processors and 4GB of RAM(64-bit), but it's adequate for small and medium-sized businesses. There is no limit on the number of users or database size. This edition of SQL Server was initially introduced to compete with lower-end vendors such as MySQL, and should be used for small organizations or departmental applications. It is easily upgraded to the other, more scalable, editions.

Web Edition (32-bit and 64-bit)

The Web Editions of SQL Server are low cost options intended for web site owners or web hosting companies. These editions include the scalability and manageability features in SQL Server 2008.

Standard Edition (32-bit and 64-bit)

The Standard Edition of SQL Server contains high availability clustering features as well as business intelligence. It is intended for small to medium-sized businesses and departmental solutions.

Enterprise, Evaluation, and Developer Editions (32-bit and 64-bit)

Enterprise Edition is the best option for SQL Server if you need to use some of the more advanced business intelligence features or if the uptime of your database is very important. Although the Standard Edition of SQL Server enables you to have high-availability options, Enterprise Edition far outdoes its sister edition with higher-end clustering as well as more advanced mirroring and log-shipping options. The counter to this, of course, is cost. This edition of SQL Server will cost you about $25,000 per processor if you choose that licensing model. (We discuss licensing later in this chapter.)

The Evaluation Edition of SQL Server is a variant of SQL Server Enterprise Edition that expires after 180 days. After the allotted evaluation period, SQL Server will no longer start. This edition has the same features as the Enterprise Edition and may be upgraded for production use. It is not licensed for production use.

The Developer Edition of SQL Server is intended for development and testing of applications using SQL Server. It contains all of the features of the Enterprise Edition. This edition is not licensed for production use.

Operating System

The edition of SQL Server that you can install varies widely based on the operating system on your server or workstation, as summarized in the following table. The table is representative and does not include each version and service pack for each OS and SQL combination which are supported.

Operating System	SQL Express	Workgroup	Web	Standard	Developer	Enterprise
Windows Server 2003 Standard (with SP2+)	✓	✓	✓	✓	✓	✓
Windows Server 2003 Enterprise (with SP2+)	✓	✓	✓	✓	✓	✓
Windows Server 2008 Standard	✓	✓	✓	✓	✓	✓
Windows Server 2008 Enterprise	✓	✓	✓	✓	✓	✓
Windows Server 2008 Data Center	✓	✓	✓	✓	✓	✓
Windows 2008 Server Data Center	✓	✓	✓	✓	✓	✓
Windows Vista	✓	✓	✓	✓	✓	✓
Windows XP Professional Edition (SP2+)l	✓	✓	✓	✓	✓	

SQL Server 2008 will not run with any of the Windows Server 2008 Core Installation options because the Windows 2008 Server Core does not support the .NET Framework, which is required by SQL Server 2008. Microsoft may add this support in a future release.

Maximum Capacity of SQL Server

Memory and the number of processors is a huge contributing factor when you're scaling SQL Server. As you can imagine, the amount of memory you can scale and the number of processors will vary based on

the edition of SQL Server you purchase. In some cases, your scalability is restricted only to the operating system's maximum memory or number of processors. This is where 64-bit becomes really useful. (We cover 64-bit scalability in much more detail in Chapter 15.)

Capacity	SQL Express	Workgroup	Web	Standard	Enterprise
Memory Supported 32-bit	1GB	OS Maximum	OS Maximum	OS Maximum	OS Maximum
Memory Supported 64-bit	N/A	4GB	OS Maximum	OS Maximum	OS Maximum
Maximum Database Size	4GB	No Limit	No Limit	No Limit	No Limit
Number of Processors	1	2	4	4	OS Maximum

Database Features by Edition

The main advantage offered by the higher (and more expensive) editions of SQL Server is the greater number of features available. In the following set of grids, you can see how the features line up across the various editions. These grids do not capture all the features of SQL Server but focus on those features of high consumer interest and areas that help distinguish the editions. This information was obtained from Microsoft Books Online.

Scalability

As demand for database resources increases, the ability to provide that higher scalability becomes very important. This list shows the scalability features, and as you might expect, they are all included in the Enterprise Edition only.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Partitioning						✓
Data compression						✓
Resource governor						✓
Partition table parallelism						✓

High Availability

Keeping your data online and ready to use is of primary importance to most facilities. These are the functions and features associated with high availability.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Multi-instance support	16	16	16	16	16	50
Online system changes	✓	✓	✓	✓	✓	✓
Backup log shipping			✓	✓	✓	✓
Database mirroring	Witness only	Witness only	Witness only	Witness only	✓ (safety full only)	✓ (full)
Failover clustering					2 nodes	OS maximum
Dynamic AWE					✓	✓
Failover without client configuration					✓	✓
Automatic corruption recovery from mirror					✓	✓
Database snapshots						✓
Fast recovery						✓
Online indexing						✓
Online restore						✓
Mirrored backups						✓
Hot add memory						✓
Online configuration of P2P nodes						✓
Hot add CPU						✓
Backup compression						✓

Security

As more data governance, auditability, and accountability is imposed, security features become more important. SQL Server 2008 included auditing, and new encryption capabilities which help meet those requirements.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
C2 compliant tracing	✓	✓	✓	✓	✓	✓
SQL auditing foundation	✓	✓	✓	✓	✓	✓
Fine grained auditing					✓	✓
Transparent database encryption						✓
ISV encryption (off-box key management)						✓

Replication

SQL Server allows you to make copies of data using replication. Depending on your data needs, you may choose periodic snapshots, transaction based replication, or replication for occasionally connected users.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Merge replication	Subscriber only	Subscriber only	Subscriber only	Subscriber only	✓	✓
Transactional replication	Subscriber only	Subscriber only	Subscriber only	Subscriber only	✓	✓
Snapshot replication	Subscriber only	Subscriber only	Subscriber only	Subscriber only	✓	✓
Change Tracking	✓	✓	✓	✓	✓	✓
Heterogeneous subscribers					✓	✓
Oracle publishing						✓
P2P transactional replication						✓

Manageability

While SQL Server databases have historically been easy to manage, Microsoft is adding improvements in this area to allow DBAs to easily manage larger groups of servers. Particularly interesting and important in this release are the policy-based management features.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
User instances	✓	✓				
Dedicated admin connection	✓ (Under trace flag)	✓ (Under trace flag)	✓	✓	✓	✓
Policy-based configuration	✓	✓	✓	✓	✓	✓
Policy-based management	✓	✓	✓	✓	✓	✓
Performance data collection and warehouse			✓	✓	✓	✓
Standard performance reports					✓	✓
Plan guides					✓	✓
Plan freezing for plan guides					✓	✓
Policy-based best practices					✓	✓
Multi-server policy-based management					✓	✓
Distributed partitioned views						✓
Parallel index operations						✓
Automatic query-to-indexed-view matching						✓
Parallel database backup checksum check						✓

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Database mail			✓	✓	✓	✓
Database migration tools	✓	✓	✓	✓	✓	✓

Management Tools

These are the management tools which come with each edition of SQL Server 2008. SQL Express Advanced now includes SQL Server Management Studio.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
SQL management objects (SMO)	✓	✓	✓	✓	✓	✓
SQL Configuration Manager	✓	✓	✓	✓	✓	✓
SQL CMD(command prompt tool)	✓	✓	✓	✓	✓	✓
SQL Server Management Studio	✓ (Express version)		✓ (Express version)	✓	✓	✓
SQL Profiler			✓	✓	✓	✓
SQL Server Agent			✓	✓	✓	✓
Database tuning advisor			✓	✓	✓	✓
Microsoft Operations Manager Pack			✓	✓	✓	✓

The table above indicates that the Web edition contains the Express version of SQL Server Management Studio. This is the information obtained from Microsoft Books Online. However, I am unsure that is true. If this information is critical to your decision about the Web version please consult Microsoft to get a determination.

Development Tools

Tight integration of development tools with SQL Server have gotten better through the years. Intellisense was a wonderful addition to the tools, and if you use Multidimensional Expression (MDX), the MDX editor is quite helpful.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Visual Studio Integration	✓	✓	✓	✓	✓	✓
SQL query, edit, and design tools				✓	✓	✓
Intellisense(Transact-SQL and MDX)				✓	✓	✓
Version control support				✓	✓	✓
Business Intelligence Development Studio					✓	✓
MDX edit, debug and design tools					✓	✓

Programmability

While notification services goes away in this release, service broker remains. Stronger XML support is also included in all editions. The new date/time data types, merge/upsert, and filestream support are also exciting new additions.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Common language runtime (CLR) integration	✓	✓	✓	✓	✓	✓
Native XML support	✓	✓	✓	✓	✓	✓
XML indexing	✓	✓	✓	✓	✓	✓
MERGE and UPSERT capabilities	✓	✓	✓	✓	✓	✓
FILESTREAM support	✓	✓	✓	✓	✓	✓
Date and Time data types	✓	✓	✓	✓	✓	✓
Internationalization support	✓	✓	✓	✓	✓	✓
Full-text search	✓		✓	✓	✓	✓

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Specification of language in query	✓		✓	✓	✓	✓
Service broker(messaging)	Client only	Client only	Client only	✓	✓	✓
XML/A support					✓	✓
Web services (HTTP/SOAP endpoints)					✓	✓

Spatial and Location Services

SQL Server 2008 has added geospatial libraries and data types, included with all editions.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Spatial indexes	✓	✓	✓	✓	✓	✓
Geodetic data type	✓	✓	✓	✓	✓	✓
Advanced spatial libraries	✓	✓	✓	✓	✓	✓
Standards-based spatial support	✓	✓	✓	✓	✓	✓

Integration Services

Integration Services allows you to extract, transform, and load data from one data source to another. Standard and Enterprise editions come with additional connectivity and transformation capabilities.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Import/Export Wizard with basic sources/destinations and Execute SQL task	✓	✓	✓	✓	✓	✓
Integration Services runtime	✓	✓	✓	✓	✓	✓

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Integration Services API and object model			✓	✓	✓	✓
SSIS Designer including VSTA scripting					✓	✓
Basic tasks and transformations					✓	✓
Log providers and logging					✓	✓
Data profiling tools					✓	✓
Additional sources and destinations:					✓	✓
Raw File source						
XML source						
Datareader destination						
Raw File destination						
Recordset destination						
SQL Server Compact destination						
SQL Server destination						
Advanced sources and destinations:						✓
Data Mining Query transformation						
Fuzzy Lookup and Fuzzy Grouping transformations						
Term Extraction and Term Lookup transformations						
Data Mining Model Training destination						
Dimension Processing destination						
Partition Processing destination						

Data Warehouse Creation

New designers and auto-generation of staging schemas, new to 2008 are included in the Standard and Enterprise edition.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Create cubes without a database					✓	✓
Auto-generate staging and data warehouse schema					✓	✓
Attribute relationship designer					✓	✓
Efficient aggregation designers					✓	✓

Data Warehouse Scale and Performance

As you might imagine, all of the performance and high scalability features are in the Enterprise edition. Change data capture is very exciting.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Change data capture						✓
Star join query optimization						✓
Scalable read-only AS configuration						✓
Proactive caching						✓
Auto parallel partition processing						✓
Partitioned cubes						✓
Distributed partitioned cubes						✓

Multi-Dimensional Analytics

Special aggregations and intelligence, and semi-additive measures are available. General performance improvements are included everywhere SSAS is supported.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
SQL Server Analysis Services service					✓	✓
SQL Server Analysis Services backup					✓	✓
General performance/scale improvements					✓	✓
Dimension, attribute, relationship, aggregate and cube design improvements					✓	✓
Personalization extensions					✓	✓
Financial aggregations						✓
Partitioned customers						✓
Custom rollups						✓
Semi-additive measures						✓
Writeback dimensions						✓
Linked measures and dimensions						✓
Binary and compressed XML transport						✓
Account intelligence						✓
Perspectives						✓
Analysis Services shared, scalable databases						✓

Data Mining

Serious data-mining efforts will require the Enterprise edition of SQL Server.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Standard algorithms					✓	✓
Data mining tools: wizards, editors, query builders					✓	✓
Cross validation						✓
Models on filtered subsets of mining structure data						✓
Time series: custom blending between ARTXP and ARIMA models						✓
Time series: prediction with new data						✓
Unlimited concurrent data mining queries						✓
Advanced configuration and tuning for algorithms						✓
Algorithm plug-in API						✓
Parallel model processing						✓
Time-series: cross-series prediction						✓
Unlimited attributes for association rules						✓
Sequence prediction						✓
Multiple prediction targets for naïve bayes, neural network and logistic regression						✓

Reporting

Reporting Services (SSRS), supported in many environments is one of the most popular SQL Server features. Particularly helpful in SQL Server 2008 is the ability to run SSRS Service outside of IIS.

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Report server	✓		✓	✓	✓	✓
Report Designer	✓		✓	✓	✓	✓
Report Manager	✓		✓	✓	✓	✓
Role-based security	✓ (Fixed roles)		✓ (Fixed roles)	✓ (Fixed roles)	✓	✓
Ad-hoc reporting (Report Builder)				✓	✓	✓
Word export and enhanced text formatting	✓		✓	✓	✓	✓
Enterprise-scale reporting engine	✓		✓	✓	✓	✓
IIS-agnostic report deployment	✓		✓	✓	✓	✓
Updated management tools	✓		✓	✓	✓	✓
Report definition customization extensions(RDCE)	✓		✓	✓	✓	✓
SharePoint integration					✓	✓
Enhanced SSRS gauges and charting	✓		✓	✓	✓	✓
Custom authentication			✓	✓	✓	✓
Export to Excel, PDF, and images	✓		✓	✓	✓	✓
Remote and non-relational data source support					✓	✓
E-mail and file share delivery					✓	✓

Feature	Express Advanced	Express	Web	Workgroup	Standard	Enterprise
Report history, scheduling, subscriptions and caching					✓	✓
Data source, delivery and rendering extensibility					✓	✓
Scale out (Web farms)						✓
Infinite click-through						✓
Data-driven subscriptions						✓
Reporting Services memory limits	4GB		4GB	4GB	Unlimited	Unlimited

Licensing

Every DBA has probably received a dreaded licensing question or two, and we hope to answer some of those common questions in this section. There is no difference in price between 32-bit and 64-bit servers. SQL Server licensing applies to all components, and to each component separately.

If you have a license on a machine, you may install all of SQL Server, Reporting Services, the SQL Server Engine, Analysis Services, and Integration Services. There are no extra licenses involved; it is a single product. However, if you wish to install only Analysis Services, or any other part of SQL Server on a difference physical server, that is another license.

Licensing can become very complicated, and licensing options are subject to change, so before you make any big commitments, it is wise to consult your Microsoft representative for an official opinion.

Licensing Options

The following list details the basic licensing models available to you:

- **Processor Licensing Model:** With the Processor licensing model, you pay a license fee for each processor. This model is a good choice if your SQL Server will be accessible over the Internet, or a high number of clients need to access each SQL Server. Using this model, there are no extra licensing fees to pay for each client or for Internet access, or anything else.
- **Server plus Device Client Access License (CAL):** The Server + Device CAL model requires you to pay a license fee for each server (instead of per processor in the processor model), plus

a license fee for each device (client computer) that accesses the SQL Server functionality. This model is good when you have a small number of clients that access SQL Server, inside the firewall, and when there are many users accessing the same device. An example of multiple users on a device might be a kiosk in a shopping center, or a call center that runs around the clock. For example, suppose each device in the call center is used by three people, one person on each of three shifts. You can buy a server license for the SQL Server plus a single CAL for the device, and all three of your employees have access. Once a device has a CAL, that device may access any licensed SQL Server in your environment.

- **Server plus User CAL:** This model requires you to pay a license fee for each server, plus a license fee for each user. This model is commonly used when your users do not need SQL access outside of the firewall and when the ratio of users to servers is small. Depending on current pricing, as of this writing User CALs might be a good option when there are fewer than 25 users per processor for the Standard edition and fewer than 75 users per processor in the Enterprise Edition. User CALs might also be a good option, compared to the device CALs, when a user has multiple devices that need access to SQL Server. This might occur when a single user has both a desktop PC and a laptop.
- **Middleware, Transaction Servers, and Multi-tiered Architectures:** The device CAL must be obtained for each *unique* device which accesses SQL Server. You may have a multi-tiered environment where the data-access layer runs on a single device, and your application, which supports many users, connects to the data-access layer, which in turn connects to SQL Server. Paying the license fee for the single data-access layer will not suffice. You must still pay a CAL for each device (PC) that the users have. Using middleware, transaction servers, or multi-tiered architecture does not enable you to avoid paying the license fee for each device or user.
- **Hyper-Threading and Multicore Processors:** There are no extra or special charges for Hyper-Threaded or multicore processors. Although you can configure SQL Server to use a lower number of processors than are installed, you will have to pay for each processor on the motherboard, as long as the operating system can see it. As an example, if you have four processors on your server, and have configured SQL Server to use only two, you will still have to pay license fees for four processors. The only way around this is to make some of the processors unavailable to the operating system by disabling them, taking them off of the motherboard, or running SQL Server in a virtual environment with fewer processors.

Virtual Server Licensing Issues

You may run SQL Server 2008 in a virtual environment. At least one license is required for SQL in each virtual environment where SQL Server runs.

- **Server/CAL License Model:** The Standard and Workgroup editions require a Server license for each instance of SQL Server running in a physical or virtual environment. For a SQL Server Standard instance in each of three virtual machines, you would need a Server license for each instance — a total of three Server licenses.

  If you are using the Enterprise Edition, you must have a Server license for each physical environment in which SQL Server runs. An example of this would include taking a large machine and partitioning it into several physical environments. Each physical environment running an instance of SQL Server Enterprise Edition requires a separate Server license. However, once one

Enterprise license exists for the physical server, there are no additional licensing requirements for other instances running within virtual environments on this physical server.

To recap, an Enterprise Edition requires a single license for each physical server, and includes all virtual servers. The Standard and Workgroup edition require a separate license for each virtual server.

- **Processor License Model:** Once a processor has a SQL Server license, it may be used for as many instances of SQL Server as you wish, in either the physical or virtual environments.

  If you run SQL in a physical environment, you must have licenses for all processors. If you run SQL in a virtual environment, you must have licenses for each processor available to the environment. You pay only once for the license for a processor, however. Suppose you have a four-processor machine and you wish to pay for licenses for two of the four processors, processors 2 and 3. You may not run SQL Server on the physical environment because you have not licensed all four of the processors. However, you may set up as many virtual environments which use only processors 2 and 3 as you wish. You may also install as many named instances of SQL Server as you wish on any of the virtual machines, because you have paid the processor license for SQL Server for the processors available to these virtual machines.

  You will never need more SQL Server processor licenses than there are processors on the server, regardless of the number of virtual machines or SQL Server instances.

- **Passive Server Licensing:** SQL Server has three features that can be used for failover support:

  - Database mirroring
  - Failover clustering
  - Backup log shipping

    These may be used to allow another server to pick up the work from a failed server. The failover or passive server is not being used by clients, except when the primary server has failed. The passive server does not have to be licensed, unless you are using the Processor licensing model, and the passive server has more processors than the active server. In this case, you must purchase processor licenses for each of the extra processors on the passive server. The passive server may take over the duties of the active server for 30 days. Afterward, the passive server must be licensed.

- **Reporting Services Licensing:** Reporting Services consists of two main pieces — the Reporting Services Web Service and the Reporting Services metadata database. The metadata database stores information about the reports, security, and subscriptions. Every Reporting Services install must connect to a SQL Server metadata database. Each of the two components must have a valid SQL license. If they are installed on the same server, you need one license. One of the first things you might do to increase scalability is to place the Reporting Services database on a separate physical server than the Reporting Services Web service — that is two licenses.
- **Processor License Model:** This model is required if you are using Reporting Services in an extranet or intranet environment. No additional Device or User CALs are necessary.
- **Server/Cal License Model:** This model requires a Server license for Reporting Services, and a device or user CAL for each device/user accessing a report, either directly on indirectly, including Report Builder and Report Designer.

Summary

In this chapter, we covered the basic architecture for SQL Server, including how it stores its data and how it communicates. We also addressed the various data types and when to use one type rather than another. Last, we answered some of the many dreaded SQL Server edition and licensing questions that we often hear from users. Now that you have the groundwork down, you're ready to jump into installing SQL Server and some of the common issues that you may experience.